**DOKUMEN DESAIN PERANGKAT LUNAK**

**HOME AUTOMATION**

SOURCE CODE

*LSKK-SW-SC-01-HOMEAUTO*



**PT. LANGGENG SEJAHTERA KREASI KOMPUTASI**

**©2019**

Dokumen Desain Perangkat Lunak, HOMEAUTOMATION

Prototype Sistem Homeautomation

PT. Langgeng Sejahtera Kreasi Komputasi ©2019

# Deskripsi Aplikasi

Sistem homeauto ini memungkinkan implementasi berbagai fitur canggih dengan menggunakan konsep IOT yang memungkinkan berbagai hal menarik muncul, seperti:

1. proses instalasi peralatan tambahan menjadi lebih cepat dan mudah;
2. berbagai fitur optimasi dapat dibuat (enabler factor);
3. berbagai fitur baru menjadi pola instrumentasi di rumah menjadi sangat berbeda (*transformative factor*).

Sistem ini merupakan salah satu sistem paling kompleks yang dibangun bukan sekedar untuk memenuhi pola instrumentasi otomatis dari sistem yang ada di rumah/bangunan, namun mencakup dari pemikiran sejak proses desain, instalasi, evolusi rumah dan proses rancangbangun nya.

# *Source code*

Berikut *source code* untuk sistem perangkat lunak penerima dan pengolah suara awal.

|  |
| --- |
| /\*  Develop by : Nurman Hariyanto  Email : nurman.hariyanto13@gmail.com  Project : Homeautomation  Version : 3.0  \*/  //library  /\*  Library  Version:  <ESP8266WiFi.h> 1.0.0  <DNSServer.h> 1.1.0  <ESP8266WebServer.h> 1.0.0  <WiFiManager.h> 0.12.0  <ArduinoJson.h> 5.13.2  <PubSubClient.h> 2.6.0  \*/  #include <FS.h> //this needs to be first, or it all crashes and burns...  #include <Ticker.h>  #include <ESP8266WiFi.h>  #include <DNSServer.h>  #include <ESP8266WebServer.h>  #include <WiFiManager.h>  #include <ArduinoJson.h>  //#include <SPI.h>  //#include "MFRC522.h"  #include <PubSubClient.h>  #include "config.h"  const int pinHigh = D1;  const int pinMedium = D2;  const int pinLow = D3;  const int Buzzer = D4;  // variables will change:  //int sensorApiState = 0;  int statusPin[3] = {0, 0, 0};  String sensorStatus[4] = {"A0", "B0", "C0","D0"};  String typeDevice = "Water Level";  /\*  Setup pin and load config file  \*/  void setup() {  pinMode(pinHigh, INPUT);  pinMode(pinMedium, INPUT);  pinMode(pinLow, INPUT);  pinMode(Buzzer, OUTPUT);  digitalWrite(pinHigh, HIGH);  digitalWrite(pinMedium, HIGH);  digitalWrite(pinLow, HIGH);  digitalWrite(Buzzer, LOW);  Serial.begin(115200);  Serial.println(F("Booting...."));  //read config wifi,mqtt dan yang lain  ReadConfigFile();  setup\_wifi();  SaveConfigFile();  client.setServer(mqtt\_server, mqtt\_port);  client.setCallback(callback);  delay(100);  }  //setup wifi  void setup\_wifi() {  WiFi.macAddress(MAC\_array);  for (int i = 0; i < sizeof(MAC\_array) - 1; ++i) {  sprintf(MAC\_char, "%s%02x:", MAC\_char, MAC\_array[i]);  }  sprintf(MAC\_char, "%s%02x", MAC\_char, MAC\_array[sizeof(MAC\_array) - 1]);  WiFiManagerParameter custom\_mqtt\_server("server", "mqtt server", mqtt\_server, 40);  String(mqtt\_port).toCharArray(smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_port("port", "mqtt port", smqtt\_port, 5);  // WiFiManagerParameter custom\_mqtt\_user("user", "mqtt user", mqtt\_user, 40);  //WiFiManagerParameter custom\_mqtt\_password("password", "mqtt password", mqtt\_password, 40);  WiFiManagerParameter custom\_mqtt\_keywords1("keyword1", "mqtt keyword1", mqtt\_keywords1, 40);  //WiFiManagerParameter custom\_mqtt\_keywords2("keyword2", "mqtt keyword2", mqtt\_keywords2, 40);  WiFiManager wifiManager;  wifiManager.setSaveConfigCallback(saveConfigCallback);  wifiManager.addParameter( & custom\_mqtt\_server);  wifiManager.addParameter( & custom\_mqtt\_port);  //wifiManager.addParameter( & custom\_mqtt\_user);  // wifiManager.addParameter( & custom\_mqtt\_password);  wifiManager.addParameter(&custom\_mqtt\_keywords1);  // wifiManager.addParameter( & custom\_mqtt\_keywords2);  //fetches ssid and pass and tries to connect  //if it does not connect it starts an access point with the specified name  //here "AutoConnectAP"  //and goes into a blocking loop awaiting configuration  if (!wifiManager.autoConnect(MAC\_char, "password")) {  Serial.println("failed to connect and hit timeout");  delay(2000);  //reset and try again, or maybe put it to deep sleep  ESP.reset();  delay(2000);  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  //calback  void callback(char \* topic, byte \* payload, unsigned int length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  Serial.println();  }  void reconnect() {  // Loop until we're reconnected  while (!client.connected()) {  Serial.print("Attempting MQTT connection...%s");  Serial.println(mqtt\_server);  // Attempt to connect  if (client.connect(MAC\_char)) {  Serial.println("connected");  } else {  Serial.print("failed, rc=");  Serial.print(client.state());  if (client.state() == 4) ESP.restart();  else {  Serial.println(" try again in 5 seconds");  }  }  }  }  //loop publish dan reconnect jika terputus dari mqtt server  void loop() {  if (!client.connected()) {  reconnect();  }  String routeKey = String(mqtt\_keywords1);  statusPin[0] = digitalRead(pinHigh);  statusPin[1] = digitalRead(pinMedium);  statusPin[2] = digitalRead(pinLow);  char dataStatusHigh[300];  char dataStatusMedium[300];  char dataStatusLow[300];  char dataStatusveryLow[300];  //test print serial  Serial.println(statusPin[0]);  //test print route key  Serial.println(routeKey);  String typeDevice = "Water Level";  String macAdd = MAC\_char;  String data2 = "0";  String data3 = "0";  String data4 = "0";  String data5 = "0";  String kodeDevice = "123456";    //pompa nyala  if ((statusPin[0] == HIGH) && (statusPin[1] == HIGH) && (statusPin[2] == HIGH))  {  digitalWrite(Buzzer, HIGH);  sensorStatus[3] = "D1";  String checkveryLow = String("#" + typeDevice + "#" +macAdd + "#" + sensorStatus[3] + "#" + data2 + "#" + data3 + "#" + data4 + "#" + data5 + "#" + kodeDevice);  checkveryLow.toCharArray(dataStatusveryLow, sizeof(dataStatusveryLow));  Serial.println(dataStatusveryLow);  if (client.publish(mqtt\_keywords1, dataStatusveryLow) == true) {  Serial.println("Success sending message");  Serial.println(dataStatusveryLow);  } else {  Serial.println("Error sending message");  }  delay(1000);  }  //switch 3  if ((statusPin[0] == HIGH) && (statusPin[1] == HIGH) && (statusPin[2] == LOW))  {  sensorStatus[2] = "C1";  String checkLow = String("#" + typeDevice + "#" +macAdd + "#" + sensorStatus[2] + "#" + data2 + "#" + data3 + "#" + data4 + "#" + data5 + "#" + kodeDevice);  checkLow.toCharArray(dataStatusLow, sizeof(dataStatusLow));  Serial.println(dataStatusLow);  if (client.publish(mqtt\_keywords1, dataStatusLow) == true) {  Serial.println("Success sending message");  Serial.println(dataStatusLow);  } else {  Serial.println("Error sending message");  }  delay(1000);  }  //switch 2  if ((statusPin[0] == HIGH) && (statusPin[1] == LOW) && (statusPin[2] == LOW))  {  sensorStatus[1] = "B1";  String checkMedium = String("#" + typeDevice + "#" +macAdd + "#" + sensorStatus[1] + "#" + data2 + "#" + data3 + "#" + data4 + "#" + data5 + "#" + kodeDevice);  checkMedium.toCharArray(dataStatusMedium, sizeof(dataStatusMedium));  Serial.println(dataStatusMedium);  if (client.publish(mqtt\_keywords1, dataStatusMedium) == true) {  Serial.println("Success sending message");  Serial.println(dataStatusMedium);  } else {  Serial.println("Error sending message");  }  delay(1000);  }  //switch 1  if ((statusPin[0] == LOW) && (statusPin[1] == LOW) && (statusPin[2] == LOW))  {  digitalWrite(Buzzer, HIGH);  sensorStatus[0] = "A1";  String checkHigh = String("#" + typeDevice + "#" +macAdd + "#" + sensorStatus[0] + "#" + data2 + "#" + data3 + "#" + data4 + "#" + data5 + "#" + kodeDevice);  checkHigh.toCharArray(dataStatusHigh, sizeof(dataStatusHigh));  Serial.println(dataStatusHigh);  if (client.publish(mqtt\_keywords1, dataStatusHigh) == true) {  Serial.println("Success sending message");  Serial.println(dataStatusHigh);  } else {  Serial.println("Error sending message");  }    }  delay(5000);  } |

Source 1. *Microcontroller* water level main

|  |
| --- |
| #ifndef \_\_CONFIGFILE\_\_  #define \_\_CONFIGFILE\_\_  //flag for saving data  bool shouldSaveConfig = false;  char wifi\_ssid[34] = "RuangIOT";  char wifi\_password[34] = "lskkiot2019";  char mqtt\_server[40] = "192.168.1.40";  int mqtt\_port = 1883;  char smqtt\_port[5] = "1883";  char mqtt\_keywords1[40] = "pubwater1"; //publish  const long interval = 2000;  uint8\_t MAC\_array[6];  char MAC\_char[18];  WiFiClient espClient;  PubSubClient client(espClient);  void SaveConfigCallback ();  void ReadConfigFile();  #endif |

Source 2. *Microcontroller* water level config

|  |
| --- |
| #include "config.h"  //callback notifying us of the need to save config  void saveConfigCallback () {  Serial.println("Should save config");  shouldSaveConfig = true;  }  void ReadConfigFile()  {  //read configuration from FS json  Serial.println("mounting FS...");  if (SPIFFS.begin()) {  Serial.println("mounted file system");  if (SPIFFS.exists("/config.json")) {  //SPIFFS.remove("/config.json");  //file exists, reading and loading  Serial.println("reading config file");  File configFile = SPIFFS.open("/config.json", "r");  if (configFile) {  Serial.println("opened config file");  size\_t size = configFile.size();  // Allocate a buffer to store contents of the file.  std::unique\_ptr<char[]> buf(new char[size]);  configFile.readBytes(buf.get(), size);  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.parseObject(buf.get());  json.printTo(Serial);  if (json.success()) {  Serial.println("\nparsed json");  strcpy(mqtt\_server, json["mqtt\_server"]);  String port = json["mqtt\_port"];  mqtt\_port = port.toInt();  strcpy(mqtt\_keywords1, json["mqtt\_keywords1"]);  } else {  Serial.println("failed to load json config");  }  }  }  } else {  Serial.println("failed to mount FS");  }  //end read  }  String IpAddress2String(const IPAddress& ipAddress)  {  return String(ipAddress[0]) + String(".") +\  String(ipAddress[1]) + String(".") +\  String(ipAddress[2]) + String(".") +\  String(ipAddress[3]) ;  }  void SaveConfigFile()  {    //save the custom parameters to FS  if (shouldSaveConfig) {  Serial.println("saving config");  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.createObject();  json["mqtt\_server"] = mqtt\_server;  json["mqtt\_port"] = String(mqtt\_port);  json["mqtt\_keywords1"] = mqtt\_keywords1;  File configFile = SPIFFS.open("/config.json", "w");  if (!configFile) {  Serial.println("failed to open config file for writing");  }  json.printTo(Serial);  json.printTo(configFile);  configFile.close();  //end save  }  }  void ResetSettings()  {  WiFiManager wifiManager;  wifiManager.resetSettings();  delay(2000);  ESP.restart();  } |

Source 3. *Microcontroller* water level configfile

|  |
| --- |
| /\*  Develop by : Nurman Hariyanto  Email : nurman.hariyanto13@gmail.com  Project : Homeautomation  Version : 4.0  ID Device : A010  \*/  /\*  Library  Version:  <ESP8266WiFi.h> 1.0.0  <DNSServer.h> 1.1.0  <ESP8266WebServer.h> 1.0.0  <WiFiManager.h> 0.12.0  <ArduinoJson.h> 5.13.2  <PubSubClient.h> 2.6.0  \*/  #include <FS.h> //this needs to be first, or it all crashes and burns...  #include <Ticker.h>  #include <ESP8266WiFi.h>  #include <DNSServer.h>  #include <ESP8266WebServer.h>  #include <WiFiManager.h>  #include <ArduinoJson.h>  //#include <SPI.h>  //#include "MFRC522.h"  #include <PubSubClient.h>  #include "config.h"  /\*  Variable pin  \*/  String statusDevice[4] = {"A0", "B0", "C0", "D0"};  int lamp1 = D1;  int lamp2 = D2;  int lamp3 = D3;  int lamp4 = D4;  //var untuk userID  String idDevice = "A010";  String jenisIot = "lampu";  /\*  Setup pin and load config file  \*/  void setup() {  //setup pin  pinMode(D1, OUTPUT);  pinMode(D2, OUTPUT);  pinMode(D3, OUTPUT);  pinMode(D4, OUTPUT);  pinMode(D5, OUTPUT);  pinMode(D6, OUTPUT);  pinMode(D7, OUTPUT);  pinMode(D8, OUTPUT);  digitalWrite(D1, LOW);  digitalWrite(D3, LOW);  digitalWrite(D4, LOW);  digitalWrite(D5, LOW);  digitalWrite(D7, LOW);  digitalWrite(D8, LOW);  Serial.begin(115200);  Serial.println(F("Booting...."));  ReadConfigFile();  setup\_wifi();  SaveConfigFile();  client.setServer(mqtt\_server, mqtt\_port);  client.setCallback(callback);  }  /\*  Configure wifi and mqtt address  \*/  void setup\_wifi() {  WiFi.macAddress(MAC\_array);  for (int i = 0; i < sizeof(MAC\_array) - 1; ++i) {  sprintf(MAC\_char, "%s%02x:", MAC\_char, MAC\_array[i]);  }  sprintf(MAC\_char, "%s%02x", MAC\_char, MAC\_array[sizeof(MAC\_array) - 1]);  //String id\_Device = "A002";  WiFiManagerParameter custom\_mqtt\_server("server", "mqtt server", mqtt\_server, 40);  String(mqtt\_port).toCharArray(smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_port("port", "mqtt port", smqtt\_port, 5);  //WiFiManagerParameter custom\_mqtt\_user("user", "mqtt user", mqtt\_user, 40);  //WiFiManagerParameter custom\_mqtt\_password("password", "mqtt password", mqtt\_password, 40);  WiFiManagerParameter custom\_mqtt\_keywords1("keyword1", "mqtt keyword1", mqtt\_keywords1, 40);  WiFiManagerParameter custom\_mqtt\_keywords2("keyword2", "mqtt keyword2", mqtt\_keywords2, 40);  WiFiManager wifiManager;  wifiManager.setSaveConfigCallback(saveConfigCallback);  wifiManager.addParameter( & custom\_mqtt\_server);  wifiManager.addParameter( & custom\_mqtt\_port);  //wifiManager.addParameter( & custom\_mqtt\_user);  //wifiManager.addParameter( & custom\_mqtt\_password);  wifiManager.addParameter(&custom\_mqtt\_keywords1);  wifiManager.addParameter( & custom\_mqtt\_keywords2);  //fetches ssid and pass and tries to connect  //if it does not connect it starts an access point with the specified name  //here "AutoConnectAP"  //and goes into a blocking loop awaiting configuration  if (!wifiManager.autoConnect(MAC\_char, "password")) {  Serial.println("failed to connect and hit timeout");  delay(2000);  //reset and try again, or maybe put it to deep sleep  ESP.reset();  delay(2000);  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  /\*  Callback to receive payload message from mqtt  \*/  void callback(char \* topic, byte \* payload, unsigned int length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  Serial.println();  if (!strncmp((char \*)payload, "A1", length)) {  digitalWrite(lamp1, HIGH);  statusDevice[0] = "A1";  }  if (!strncmp((char \*)payload, "A0", length)) {  digitalWrite(lamp1, LOW);  statusDevice[0] = "A0";  }  if (!strncmp((char \*)payload, "B1", length)) {  digitalWrite(lamp2, HIGH);  statusDevice[1] = "B1";  }  if (!strncmp((char \*)payload, "B0", length)) {  digitalWrite(lamp2, LOW);  statusDevice[1] = "B0";  }  if (!strncmp((char \*)payload, "C1", length)) {  digitalWrite(lamp3, HIGH);  statusDevice[2] = "C1";  }  if (!strncmp((char \*)payload, "C0", length)) {  digitalWrite(lamp3, LOW);  statusDevice[2] = "C0";  }  if (!strncmp((char \*)payload, "D1", length)) {  digitalWrite(lamp4, HIGH);  statusDevice[3] = "D1";  }  if (!strncmp((char \*)payload, "D0", length)) {  digitalWrite(lamp4, LOW);  statusDevice[3] = "D0";  }  if (!strncmp((char \*)payload, "E1", length)) {  digitalWrite(lamp1, HIGH);  digitalWrite(lamp2, HIGH);  digitalWrite(lamp3, HIGH);  digitalWrite(lamp4, HIGH);  statusDevice[0] = "A1";  statusDevice[1] = "B1";  statusDevice[2] = "C1";  statusDevice[3] = "D1";  }  if (!strncmp((char \*)payload, "E0", length)) {  digitalWrite(lamp1, LOW);  digitalWrite(lamp2, LOW);  digitalWrite(lamp3, LOW);  digitalWrite(lamp4, LOW);  statusDevice[0] = "A0";  statusDevice[1] = "B0";  statusDevice[2] = "C0";  statusDevice[3] = "D0";  }  String data5 = "0";  String check = String("#" + jenisIot + "#" + MAC\_char + "#" + statusDevice[0] + "#" + statusDevice[1] + "#" + statusDevice[2] + "#" + statusDevice[3] + "#" + data5 + "#" + idDevice);  char dataStatus[200];  check.toCharArray(dataStatus, sizeof(dataStatus));  Serial.println(dataStatus);  if (client.publish(mqtt\_keywords1, dataStatus) == true) {  Serial.println("Success sending message");  Serial.println(dataStatus);  } else {  Serial.println("Error sending message");  }  }  /\*  Reconnect to MQTT  \*/  void reconnect() {  // Loop until we're reconnected  while (!client.connected()) {  Serial.print("Attempting MQTT connection...%s");  Serial.println(mqtt\_server);  // Attempt to connect  if (client.connect(MAC\_char)) {  Serial.println("connected");  // Once connected, publish an announcement...  // client.publish(mqtt\_keywords1, "halo smarthome");  client.subscribe(mqtt\_keywords2);  } else {  Serial.print("failed, rc=");  Serial.print(client.state());  if (client.state() == 4) ESP.restart();  else {  Serial.println(" try again in 5 seconds");  }  }  }  }  /\*  Loop  \*/  void loop() {  if (!client.connected()) {  reconnect();  }  client.loop();  }  © 2019 GitHub, Inc. |

Source 1. *Microcontroller* lampu main

|  |
| --- |
| #ifndef \_\_CONFIGFILE\_\_  #define \_\_CONFIGFILE\_\_  //flag for saving data  bool shouldSaveConfig = false;  // Update these with values suitable for your network.  // CAFE BENE Setting  /\*const char\* wifi\_ssid = "CaffeBene";  const char\* wifi\_password = "Misugaru";  const char\* mqtt\_server = "167.205.7.226";  const int mqtt\_port = 1883;  const char\* mqtt\_user = "/absensi:absensi";  const char\* mqtt\_password = "12345";\*/  // Home Setting  /\*char wifi\_ssid[34] = "Homeauto";  char wifi\_password[34] = "homeauto";  char mqtt\_server[40] = "192.168.0.2";  int mqtt\_port = 1883;  char smqtt\_port[5] = "1883";  char mqtt\_user[40] = "/absensi:absensi";  char mqtt\_password[40] = "12345";  char mqtt\_keywords1[40] = "absensimachine/feedcard";  char mqtt\_keywords2[40] = "absensimachine/control";\*/  //// PPTIK Setting  ////char wifi\_ssid[34] = "Hotspot LSKK";  ////char wifi\_password[34] = "lskkhotspot";  //char wifi\_ssid[34] = "MIC - Gigabit";  #ifndef \_\_CONFIGFILE\_\_  #define \_\_CONFIGFILE\_\_  //flag for saving data  bool shouldSaveConfig = false;  char wifi\_ssid[34] = "RuangIOT";  char wifi\_password[34] = "lskkiot2019";  char mqtt\_server[40] = "192.168.1.40";  int mqtt\_port = 1883;  char smqtt\_port[5] = "1883";  char mqtt\_keywords1[40] = "pubrelay2"; //publish  char mqtt\_keywords2[40] = "subrelay2"; //subcribe  const long interval = 2000;  uint8\_t MAC\_array[6];  char MAC\_char[18];  WiFiClient espClient;  PubSubClient client(espClient);  void SaveConfigCallback ();  void ReadConfigFile();  #endif |

Source 2. *Microcontroller* lampu config

|  |
| --- |
| #include "config.h"  //callback notifying us of the need to save config  void saveConfigCallback () {  Serial.println("Should save config");  shouldSaveConfig = true;  }  void ReadConfigFile()  {  //read configuration from FS json  Serial.println("mounting FS...");  if (SPIFFS.begin()) {  Serial.println("mounted file system");  if (SPIFFS.exists("/config.json")) {  //SPIFFS.remove("/config.json");  //file exists, reading and loading  Serial.println("reading config file");  File configFile = SPIFFS.open("/config.json", "r");  if (configFile) {  Serial.println("opened config file");  size\_t size = configFile.size();  // Allocate a buffer to store contents of the file.  std::unique\_ptr<char[]> buf(new char[size]);  configFile.readBytes(buf.get(), size);  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.parseObject(buf.get());  json.printTo(Serial);  if (json.success()) {  Serial.println("\nparsed json");  strcpy(mqtt\_server, json["mqtt\_server"]);  String port = json["mqtt\_port"];  mqtt\_port = port.toInt();  // strcpy(mqtt\_user, json["mqtt\_user"]);  // strcpy(mqtt\_password, json["mqtt\_password"]);  strcpy(mqtt\_keywords1, json["mqtt\_keywords1"]);  strcpy(mqtt\_keywords2, json["mqtt\_keywords2"]);  } else {  Serial.println("failed to load json config");  }  }  }  } else {  Serial.println("failed to mount FS");  }  //end read  }  String IpAddress2String(const IPAddress& ipAddress)  {  return String(ipAddress[0]) + String(".") +\  String(ipAddress[1]) + String(".") +\  String(ipAddress[2]) + String(".") +\  String(ipAddress[3]) ;  }  void SaveConfigFile()  {    //save the custom parameters to FS  if (shouldSaveConfig) {  Serial.println("saving config");  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.createObject();  json["mqtt\_server"] = mqtt\_server;  json["mqtt\_port"] = String(mqtt\_port);  // json["mqtt\_user"] = mqtt\_user;  //json["mqtt\_password"] = mqtt\_password;  json["mqtt\_keywords1"] = mqtt\_keywords1;  json["mqtt\_keywords2"] = mqtt\_keywords2;    File configFile = SPIFFS.open("/config.json", "w");  if (!configFile) {  Serial.println("failed to open config file for writing");  }  json.printTo(Serial);  json.printTo(configFile);  configFile.close();  //end save  }  }  void ResetSettings()  {  WiFiManager wifiManager;  wifiManager.resetSettings();  delay(2000);  ESP.restart();  } |

Source 3. *Microcontroller* lampu configfile

|  |
| --- |
| /\*  Develop by : Nurman Hariyanto  Email : nurman.hariyanto13@gmail.com  Updated by : Fahmi Nurfadilah  Email : fahmi.nurfadilah1412@gmail.com  Project : Homeautomation  Version : 3.1  \*/  //library  #include <FS.h> //this needs to be first, or it all crashes and burns...  #include <Ticker.h>  #include <ESP8266WiFi.h>  #include <DNSServer.h>  #include <ESP8266WebServer.h>  #include <WiFiManager.h>  #include <ArduinoJson.h>  //#include <SPI.h>  //#include "MFRC522.h"  #include <PubSubClient.h>  #include "config.h"  #ifndef MQTT\_MAX\_PACKET\_SIZE  #define MQTT\_MAX\_PACKET\_SIZE 2000  #endif  #include <Adafruit\_GFX.h> // Include core graphics library  #include <Adafruit\_ILI9341.h> // Include Adafruit\_ILI9341 library to drive the display  #include "DHT.h"  // Declare pins for the display:  #define TFT\_DC D2  #define TFT\_LED D8 // You can also connect this to the Arduino reset in which case, set this #define pin to -1!  #define TFT\_CS D1  // The rest of the pins are pre-selected as the default hardware SPI for Arduino Uno (SCK = 13 and SDA = 11)  #define DHTPIN D0 //Pin apa yang digunakan  #define DHTTYPE DHT11 // DHT 11  // Create display:  Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC, TFT\_LED);  #include <Fonts/FreeSerif24pt7b.h> // Add a custom font  void indoorDHT11();  int Variable1; // Create a variable to have something dynamic to show on the display  //var temp untuk menampung data json dari payload  char payloadTemp[300];  String temperature;  String pressure;  String humidity;  String indoorTemp;  DHT dht(DHTPIN, DHTTYPE);  void setup() {  //setup pin  Serial.begin(115200);  dht.begin();  Serial.println(F("Booting...."));  //read config wifi,mqtt dan yang lain  ReadConfigFile();  setup\_wifi();  SaveConfigFile();  client.setServer(mqtt\_server, mqtt\_port);  client.setCallback(callback);  }  //setup wifi  void setup\_wifi() {  WiFi.macAddress(MAC\_array);  for (int i = 0; i < sizeof(MAC\_array) - 1; ++i) {  sprintf(MAC\_char, "%s%02x:", MAC\_char, MAC\_array[i]);  }  sprintf(MAC\_char, "%s%02x", MAC\_char, MAC\_array[sizeof(MAC\_array) - 1]);  WiFiManagerParameter custom\_mqtt\_server("server", "mqtt server", mqtt\_server, 40);  String(mqtt\_port).toCharArray(smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_port("port", "mqtt port", smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_user("user", "mqtt user", mqtt\_user, 40);  WiFiManagerParameter custom\_mqtt\_password("password", "mqtt password", mqtt\_password, 40);  WiFiManagerParameter custom\_mqtt\_keywords1("keyword1", "mqtt keyword1", mqtt\_keywords1, 40);  WiFiManager wifiManager;  wifiManager.setSaveConfigCallback(saveConfigCallback);  wifiManager.addParameter( & custom\_mqtt\_server);  wifiManager.addParameter( & custom\_mqtt\_port);  wifiManager.addParameter( & custom\_mqtt\_user);  wifiManager.addParameter( & custom\_mqtt\_password);  wifiManager.addParameter(&custom\_mqtt\_keywords1);  //fetches ssid and pass and tries to connect  //if it does not connect it starts an access point with the specified name  //here "AutoConnectAP"  //and goes into a blocking loop awaiting configuration  if (!wifiManager.autoConnect(MAC\_char, "password")) {  Serial.println("failed to connect and hit timeout");  delay(2000);  //reset and try again, or maybe put it to deep sleep  ESP.reset();  delay(2000);  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  //calback  void callback(char \* topic, byte \* payload, unsigned int length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  Serial.println();  for (int i = 0; i < length; i++)  {  Serial.print((char)payload[i]);  payloadTemp[i] = payload[i];  }  Serial.println();  StaticJsonBuffer < 300 > jsonBuffer;  JsonObject & parsed = jsonBuffer.parseObject(payloadTemp);  Serial.println(payloadTemp);  if (!parsed.success()) {  Serial.println("Error parsing");  }  // menampung isi object json dari payload  String dataTemp = parsed["temp"];  String dataPressure = parsed["pressure"];  String dataHumidity = parsed["humidity"];  Serial.println("ini data dari RMQ");  Serial.print("Temperature: ");  Serial.print (dataTemp);  Serial.println();  Serial.print("Pressure: ");  Serial.print (dataPressure);  Serial.println();  Serial.print("Humidity: ");  Serial.print (dataHumidity);  Serial.println();  temperature = dataTemp;  humidity = dataHumidity;  pressure = dataPressure;  //coding ke layar ini  /\*  variabel untuk temperatur --> dataTemp  variabel untuk Pressure --> dataPressure  variabel untuk Humidity --> dataHumidity  /  \*/  display();  }  void reconnect() {  // Loop until we're reconnected  while (!client.connected()) {  Serial.print("Attempting MQTT connection...%s");  Serial.println(mqtt\_server);  // Attempt to connect  if (client.connect(MAC\_char, mqtt\_user, mqtt\_password)) {  Serial.println("connected");  // Once connected, publish an announcement...  //client.publish(mqtt\_keywords1, "halo smarthome");  client.subscribe(mqtt\_keywords1);  } else {  Serial.print("failed, rc=");  Serial.print(client.state());  if (client.state() == 4) ESP.restart();  else {  Serial.println(" try again in 5 seconds");  // Wait 5 seconds before retrying  //delay(10000);  }  }  }  }  //weather indoor  void display() {  tft.begin(); // Initialize display  tft.fillScreen(0x0000); // Fill screen with black  //tft.setRotation(0); // Set orientation of the display. Values are from 0 to 3. If not declared, orientation would be 0,  // which is portrait mode.  tft.setTextWrap(false); // By default, long lines of text are set to automatically “wrap” back to the leftmost column.  // To override this behavior (so text will run off the right side of the display - useful for  // scrolling marquee effects), use setTextWrap(false). The normal wrapping behavior is restored  // with setTextWrap(true).  // We are going to print on the display everything that is static on the setup, to leave the loop free for dynamic elements:  // Write to the display the text "Hello":  tft.setCursor(0, 0); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(3); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.print("Temperature"); // Print a text or value  tft.setCursor(0, 40); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(2); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.println(temperature); // Print a text or value  tft.setCursor(0, 80); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(3); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.println("Pressure"); // Print a text or value  tft.setCursor(0, 120); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(2); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.println(pressure); // Print a text or value  tft.setCursor(0, 160); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(3); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.println("Humidity"); // Print a text or value  tft.setCursor(0, 200); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(2); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.println(humidity); // Print a text or value  tft.setCursor(0, 240); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(2); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.println("Indoor Temperature"); // Print a text or value    tft.setCursor(0, 280); // Set position (x,y)  tft.setTextColor(0xFFFF); // Set color of text. First is the color of text and after is color of background  tft.setTextSize(2); // Set text size. Goes from 0 (the smallest) to 20 (very big)  tft.println(dht.readTemperature()); // Print a text or value  }  //loop publish  void loop() {  // unsigned long currentMillis = millis();  if (!client.connected()) {  reconnect();  //delay(100);  }  client.loop();  //display();  //Serial.println("-------------");  } |

Source 1. *Microcontroller* weather main

|  |
| --- |
| ifndef \_\_CONFIGFILE\_\_  #define \_\_CONFIGFILE\_\_  //flag for saving data  bool shouldSaveConfig = false;  char wifi\_ssid[34] = "RuangIOT";  char wifi\_password[34] = "lskkiot2019";  char mqtt\_server[40] = "167.205.7.226";  int mqtt\_port = 1883;  char smqtt\_port[5] = "1883";  char mqtt\_user[40] = "/homeauto:homeauto";  char mqtt\_password[40] = "homeauto12345!";  char mqtt\_keywords1[40] = "topicweather"; //publish  const long interval = 2000;  uint8\_t MAC\_array[6];  char MAC\_char[18];  WiFiClient espClient;  PubSubClient client(espClient);  void SaveConfigCallback ();  void ReadConfigFile();  #endif |

Source 2. *Microcontroller* weather config

|  |
| --- |
| #include "config.h"  //callback notifying us of the need to save config  void saveConfigCallback () {  Serial.println("Should save config");  shouldSaveConfig = true;  }  void ReadConfigFile()  {  //read configuration from FS json  Serial.println("mounting FS...");  if (SPIFFS.begin()) {  Serial.println("mounted file system");  if (SPIFFS.exists("/config.json")) {  //SPIFFS.remove("/config.json");  //file exists, reading and loading  Serial.println("reading config file");  File configFile = SPIFFS.open("/config.json", "r");  if (configFile) {  Serial.println("opened config file");  size\_t size = configFile.size();  // Allocate a buffer to store contents of the file.  std::unique\_ptr<char[]> buf(new char[size]);  configFile.readBytes(buf.get(), size);  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.parseObject(buf.get());  json.printTo(Serial);  if (json.success()) {  Serial.println("\nparsed json");  strcpy(mqtt\_server, json["mqtt\_server"]);  String port = json["mqtt\_port"];  mqtt\_port = port.toInt();  strcpy(mqtt\_user, json["mqtt\_user"]);  strcpy(mqtt\_password, json["mqtt\_password"]);  strcpy(mqtt\_keywords1, json["mqtt\_keywords1"]);  } else {  Serial.println("failed to load json config");  }  }  }  } else {  Serial.println("failed to mount FS");  }  //end read  }  String IpAddress2String(const IPAddress& ipAddress)  {  return String(ipAddress[0]) + String(".") +\  String(ipAddress[1]) + String(".") +\  String(ipAddress[2]) + String(".") +\  String(ipAddress[3]) ;  }  void SaveConfigFile()  {    //save the custom parameters to FS  if (shouldSaveConfig) {  Serial.println("saving config");  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.createObject();  json["mqtt\_server"] = mqtt\_server;  json["mqtt\_port"] = String(mqtt\_port);  json["mqtt\_user"] = mqtt\_user;  json["mqtt\_password"] = mqtt\_password;  json["mqtt\_keywords1"] = mqtt\_keywords1;    File configFile = SPIFFS.open("/config.json", "w");  if (!configFile) {  Serial.println("failed to open config file for writing");  }  json.printTo(Serial);  json.printTo(configFile);  configFile.close();  //end save  }  }  void ResetSettings()  {  WiFiManager wifiManager;  wifiManager.resetSettings();  delay(2000);  ESP.restart();  } |

Source 3. *Microcontroller* weather configfile

|  |
| --- |
| /\*  Develop by : Nurman Hariyanto  Email : nurman.hariyanto13@gmail.com  Updated by : Fahmi Nurfadilah  Email : fahmi.nurfadilah1412@gmail.com  Project : Homeautomation  Version : 3.1  \*/  //library  #include <FS.h> //this needs to be first, or it all crashes and burns...  #include <Ticker.h>  #include <ESP8266WiFi.h>  #include <DNSServer.h>  #include <ESP8266WebServer.h>  #include <WiFiManager.h>  #include <ArduinoJson.h>  //#include <SPI.h>  //#include "MFRC522.h"  #include <PubSubClient.h>  #include "config.h"  //var temp untuk menampung data json dari payload  char payloadTemp[200];  String typeDevice = "Dispenser";  ////var untuk PIN  //const int devices[3] = {  // D1,  // D2,  // D3  //};  //  ////var untuk status device  //int statDevice[3] = {  // 1,  // 1,  // 0  //  //};  String statusDevice[4] = {"A0", "B0", "C0", "E0"};  String statusDeviceInput[3] = {"A0" "B0" "C0"};  int lamp1 = D1;  int lamp2 = D2;  int lamp3 = D3;  int statwater = D4;  int statHot = D5;  int statCool = A0;  //var untuk userID  String userId;  String processor = "arduino";  String typemessRelay = "Dispenser Relay";  String typemessLed = "Dispenser Led";  String idDevice = "A011";  void setup() {  //setup pin  pinMode(D1, OUTPUT);  pinMode(D2, OUTPUT);  pinMode(D3, OUTPUT);  pinMode(D4, INPUT\_PULLUP);  pinMode(D5, INPUT\_PULLUP);  pinMode(A0, INPUT);  digitalWrite(D1, HIGH);  digitalWrite(D2, HIGH);  digitalWrite(D3, HIGH);  Serial.begin(115200);  Serial.println(F("Booting...."));  //read config wifi,mqtt dan yang lain  ReadConfigFile();  setup\_wifi();  SaveConfigFile();  client.setServer(mqtt\_server, mqtt\_port);  client.setCallback(callback);  }  //setup wifi  void setup\_wifi() {  WiFi.macAddress(MAC\_array);  for (int i = 0; i < sizeof(MAC\_array) - 1; ++i) {  sprintf(MAC\_char, "%s%02x:", MAC\_char, MAC\_array[i]);  }  sprintf(MAC\_char, "%s%02x", MAC\_char, MAC\_array[sizeof(MAC\_array) - 1]);  WiFiManagerParameter custom\_mqtt\_server("server", "mqtt server", mqtt\_server, 40);  String(mqtt\_port).toCharArray(smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_port("port", "mqtt port", smqtt\_port, 5);  //WiFiManagerParameter custom\_mqtt\_user("user", "mqtt user", mqtt\_user, 40);  //WiFiManagerParameter custom\_mqtt\_password("password", "mqtt password", mqtt\_password, 40);  WiFiManagerParameter custom\_mqtt\_keywords1("keyword1", "mqtt keyword1", mqtt\_keywords1, 40);  WiFiManagerParameter custom\_mqtt\_keywords2("keyword2", "mqtt keyword2", mqtt\_keywords2, 40);  WiFiManagerParameter custom\_mqtt\_keywords3("keyword3", "mqtt keyword3", mqtt\_keywords3, 40);  WiFiManager wifiManager;  wifiManager.setSaveConfigCallback(saveConfigCallback);  wifiManager.addParameter( & custom\_mqtt\_server);  wifiManager.addParameter( & custom\_mqtt\_port);  //wifiManager.addParameter( & custom\_mqtt\_user);  //wifiManager.addParameter( & custom\_mqtt\_password);  wifiManager.addParameter(&custom\_mqtt\_keywords1);  wifiManager.addParameter( & custom\_mqtt\_keywords2);  wifiManager.addParameter( & custom\_mqtt\_keywords3);  //fetches ssid and pass and tries to connect  //if it does not connect it starts an access point with the specified name  //here "AutoConnectAP"  //and goes into a blocking loop awaiting configuration  if (!wifiManager.autoConnect(MAC\_char, "password")) {  Serial.println("failed to connect and hit timeout");  delay(10000);  //reset and try again, or maybe put it to deep sleep  ESP.reset();  delay(2000);  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  //calback  void callback(char \* topic, byte \* payload, unsigned int length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  Serial.println();  if (!strncmp((char \*)payload, "A1", length)) {  digitalWrite(lamp1, LOW);  statusDevice[0] = "A1";  }  if (!strncmp((char \*)payload, "A0", length)) {  digitalWrite(lamp1, HIGH);  statusDevice[0] = "A0";  }  if (!strncmp((char \*)payload, "B1", length)) {  digitalWrite(lamp2, LOW);  statusDevice[1] = "B1";  }  if (!strncmp((char \*)payload, "B0", length)) {  digitalWrite(lamp2, HIGH);  statusDevice[1] = "B0";  }  if (!strncmp((char \*)payload, "C1", length)) {  digitalWrite(lamp3, LOW);  statusDevice[2] = "C1";  }  if (!strncmp((char \*)payload, "C0", length)) {  digitalWrite(lamp3, HIGH);  statusDevice[2] = "C0";  }  if (!strncmp((char \*)payload, "E1", length)) {  digitalWrite(lamp1, LOW);  digitalWrite(lamp2, LOW);  digitalWrite(lamp3, LOW);  statusDevice[0] = "A1";  statusDevice[1] = "B1";  statusDevice[2] = "C1";  }  if (!strncmp((char \*)payload, "E0", length)) {  digitalWrite(lamp1, HIGH);  digitalWrite(lamp2, HIGH);  digitalWrite(lamp3, HIGH);  statusDevice[0] = "A0";  statusDevice[1] = "B0";  statusDevice[2] = "C0";  }  String datarelay4 = "0";  String datarelay5 = "0";  String routeKeyOutput = String(mqtt\_keywords1);  String check = String( "#" + typemessRelay + "#" + MAC\_char + "#" + statusDevice[0] + "#" + statusDevice[1] + "#" + statusDevice[2] + "#" + datarelay4 + "#" + datarelay5 + "#" + idDevice);  char dataStatusOutput[300];  check.toCharArray(dataStatusOutput, sizeof(dataStatusOutput));  Serial.println(dataStatusOutput);  if (client.publish(mqtt\_keywords1, dataStatusOutput) == true) {  Serial.println("Success sending message");  Serial.println(dataStatusOutput);  } else {  Serial.println("Error sending message");  }  }  void reconnect() {  // Loop until we're reconnected  while (!client.connected()) {  Serial.print("Attempting MQTT connection...%s");  Serial.println(mqtt\_server);  // Attempt to connect  if (client.connect(MAC\_char)) {  Serial.println("connected");  // Once connected, publish an announcement...  //client.publish(mqtt\_keywords1, "halo smarthome");  client.subscribe(mqtt\_keywords2);  } else {  Serial.print("failed, rc=");  Serial.print(client.state());  if (client.state() == 4) ESP.restart();  else {  Serial.println(" try again in 5 seconds");  // Wait 5 seconds before retrying  //delay(10000);  }  }  }  }  //loop publish  void loop() {  // unsigned long currentMillis = millis();  if (!client.connected()) {  reconnect();  //delay(100);  }  client.loop();  int oldstateInput[3] = {0, 0, 0};  int statWater = digitalRead(D4);  int statHot = digitalRead(D5);  int statCool = analogRead(A0);  String dataled4 = "0";  String dataled5 = "0";  String routeKeyInput = String(mqtt\_keywords3);  if (statWater == HIGH) {  statusDeviceInput[0] = "A0";    }  if (statWater == LOW) {  statusDeviceInput[0] = "A1";  }  if (statHot == HIGH) {  statusDeviceInput[1] = "B1";    }  if (statHot == LOW) {  statusDeviceInput[1] = "B0";  }  if (statCool >= 0 && statCool <= 620) {  statusDeviceInput[2] = "C0";  }  if (statCool >= 620 && statCool <= 1024) {  statusDeviceInput[2] = "C1";  }  String dataSend = String("#" + typemessLed + "#" + MAC\_char + "#" + statusDeviceInput[0] + "#" + statusDeviceInput[1] + "#" + statusDeviceInput[2] + "#" + dataled4 + "#" + dataled5 + "#" + idDevice);  char dataStatusInput[300];  dataSend.toCharArray(dataStatusInput, sizeof(dataStatusInput));  Serial.println(dataStatusInput);  if (client.publish(mqtt\_keywords3, dataStatusInput) == true) {  Serial.println("Success sending message");  Serial.println(dataStatusInput);  } else {  Serial.println("Error sending message");  }  delay(3000);  //Serial.println("-------------");  } |

Source 1. *Microcontroller* dispenser main

|  |
| --- |
| #ifndef \_\_CONFIGFILE\_\_  #define \_\_CONFIGFILE\_\_  //flag for saving data  bool shouldSaveConfig = false;  char wifi\_ssid[34] = "RuangIOT";  char wifi\_password[34] = "lskkiot2019";  char mqtt\_server[40] = "192.168.1.40";  int mqtt\_port = 1883;  char smqtt\_port[5] = "1883";  //char mqtt\_user[40] = "/homeauto:homeauto";  //char mqtt\_password[40] = "homeauto1234";  char mqtt\_keywords1[40] = "pubdispenser"; //publish  char mqtt\_keywords2[40] = "subdispenser"; //subcribe  char mqtt\_keywords3[40] = "statusdispenser"; //subcribe  const long interval = 2000;  uint8\_t MAC\_array[6];  char MAC\_char[18];  WiFiClient espClient;  PubSubClient client(espClient);  //MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Create MFRC522 instance  void SaveConfigCallback ();  void ReadConfigFile();  #endif |

Source 2. *Microcontroller* dispenser config

|  |
| --- |
| #include "config.h"  //callback notifying us of the need to save config  void saveConfigCallback () {  Serial.println("Should save config");  shouldSaveConfig = true;  }  void ReadConfigFile()  {  //read configuration from FS json  Serial.println("mounting FS...");  if (SPIFFS.begin()) {  Serial.println("mounted file system");  if (SPIFFS.exists("/config.json")) {  //SPIFFS.remove("/config.json");  //file exists, reading and loading  Serial.println("reading config file");  File configFile = SPIFFS.open("/config.json", "r");  if (configFile) {  Serial.println("opened config file");  size\_t size = configFile.size();  // Allocate a buffer to store contents of the file.  std::unique\_ptr<char[]> buf(new char[size]);  configFile.readBytes(buf.get(), size);  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.parseObject(buf.get());  json.printTo(Serial);  if (json.success()) {  Serial.println("\nparsed json");  strcpy(mqtt\_server, json["mqtt\_server"]);  String port = json["mqtt\_port"];  mqtt\_port = port.toInt();  //strcpy(mqtt\_user, json["mqtt\_user"]);  //strcpy(mqtt\_password, json["mqtt\_password"]);  strcpy(mqtt\_keywords1, json["mqtt\_keywords1"]);  strcpy(mqtt\_keywords2, json["mqtt\_keywords2"]);  strcpy(mqtt\_keywords3, json["mqtt\_keywords2"]);  } else {  Serial.println("failed to load json config");  }  }  }  } else {  Serial.println("failed to mount FS");  }  //end read  }  String IpAddress2String(const IPAddress& ipAddress)  {  return String(ipAddress[0]) + String(".") + \  String(ipAddress[1]) + String(".") + \  String(ipAddress[2]) + String(".") + \  String(ipAddress[3]) ;  }  void SaveConfigFile()  {  //save the custom parameters to FS  if (shouldSaveConfig) {  Serial.println("saving config");  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.createObject();  json["mqtt\_server"] = mqtt\_server;  json["mqtt\_port"] = String(mqtt\_port);  //json["mqtt\_user"] = mqtt\_user;  //json["mqtt\_password"] = mqtt\_password;  json["mqtt\_keywords1"] = mqtt\_keywords1;  json["mqtt\_keywords2"] = mqtt\_keywords2;  json["mqtt\_keywords3"] = mqtt\_keywords3;  File configFile = SPIFFS.open("/config.json", "w");  if (!configFile) {  Serial.println("failed to open config file for writing");  }  json.printTo(Serial);  json.printTo(configFile);  configFile.close();  //end save  }  }  void ResetSettings()  {  WiFiManager wifiManager;  wifiManager.resetSettings();  delay(2000);  ESP.restart();  } |

Source 3. *Microcontroller* dispenser configfile

# 

# *Release Notes*

Berikut *release notes* aplikasi untuk sistem Homeautomation.

### *HOMEAUTO-SC, b1912, v1.0*

* Versi rilis pertama.

# *Approval*

Dibuat oleh Tim *S/W* *Engineer* pada 13 Desember 2019.

Dicek dan disetujui pada 13 Desember, oleh:

Mochamad Vicky Ghani Aziz